Team Project I: Rubik's Cube Visualizer

Due Date: Wednesday, October 30 @ 1830

Submission Format: Electronic via iLearn

Notes: Each team shall comprise two students. Only one submission is required for each team.

Summary

This project is an extension of the Cube related assignments we have worked on thus far.

The goal of this application is to implement a Rubik's Cube visualizer that first reads in both an initial state and subsequent solution for the Rubik's Cube and then animates the solving of the Cube as a corresponding series of rotations.

Requirements

Your Cube model can be simple, but maximum points are earned by detailed or realistic shapes – for example, some small gaps between blocks, slightly rounded corners, etc.

Each step in the solution should be animated smoothly – the duration of the animation is up to you, but it should be quick enough to keep the viewer's interest. You should include controls (button, keyboard, etc.) for replaying the animation, loading a new solution file, changing the camera angle/projection, and quitting the application.

In addition to these requirements, you must also incorporate smooth, realistic shading into your application. The more attractive or realistic your object looks, the more points you will get toward this component of the application. Include at least two light sources with different properties to give your scene added depth.

Of course, coding style, modularity, and other best practices also factor into the project grade.

Grading

A breakdown of the individual components of the application and their weights is given below.

- Modeling & Materials 20%
- Transformations & Animation 20%
- Camera & Viewing 10%
- Lights & Shading 20%
- User Interaction 10%
- Read/Process Rubik Solution 10%
- Overall Design & Quality 10%

Data Representation

The initial state of the Cube is given by a text string of the following form (where the specific colors may differ):

```
   GGW
   RRG
   RRG
OWWGGOYYR
OGOYYYRBR
YYYRBGRWW
   BOY
   BOB
   BOB
   OGO
   WWB
   WWB
```

Each solution is given as a text string of the following form (this is a solution to the state above):

```
01W1R1Y3
```

Note that the center square of each side of the Cube will never move (at most only spin around its own axis), so a single step in the solution such as `01` means that we rotate the face with the *orange* square at its center. All rotations will be *clockwise* by 90 degrees.

You can use the state and solution above as test inputs, and you can also devise your own for additional testing (whether they represent real solutions does not matter for testing the visualizer).

Additional Tips

Take a look at a real Rubik's cube (either in person or via photo/video) to see how it is shaped, what the colors and reflective properties look like under real lighting conditions.

Use HTML5 and new JavaScript local I/O capabilities for reading. We will touch on this in class.

It would be most useful to work together on the planning and design up front, and then identify specific tasks to delegate to each team member after that. The outline of the program, functions, etc. may need to be rethought and reworked from the simple Cube examples we are building upon. Then, perhaps one team member tackles modeling while the other handles the shader implementation. Or perhaps one implements the user controls and the other write the camera and transform functions that will attach to those controls.

It is always a good idea to think about the problem from start to finish, spec out the necessary work, and then write a time line for your team so that you stay on task and complete the application by the deadline.